Comparison of Architectural Design Decisions for Resource-Constrained Self-Driving Cars – A Multiple Case-Study

Christian Berger[1] and Michael Dukaczewski[2]

[1]Department of Computer Science and Engineering
Chalmers | University of Gothenburg
Sweden
christian.berger@chalmers.se

[2]Software Engineering and Automotive Informatics
Technische Universität Braunschweig
Germany
m.dukaczewski@tu-bs.de

Abstract: *Context:* Self-Driving cars are getting more and more attention with public demonstration from all important automotive OEMs but also from companies, which do not have a long history in the automotive industry. Fostered by large international competitions in the last decade, several automotive OEMs have already announced to bring this technology to the market around 2020.
Objective: International competitions like the 2007 DARPA Urban Challenge did not focus on efficient usage of resources to realize the self-driving vehicular functionality. Since the automotive industry is very cost-sensitive, realizing reliable and robust self-driving functionality is challenging when expensive and sophisticated sensors mounted very visibly on the vehicle's roof for example cannot be used. Therefore, the goal for this study is to investigate how architectural design decisions of recent self-driving vehicular technology consider resource-efficiency.
Method: In a multiple case study, the architectural design decisions derived for resource-constrained self-driving miniature cars for the international competition CaroloCup are compared with architectural designs from recent real-scale self-driving cars.
Results: Scaling down available resources for realizing self-driving vehicular technology puts additional constraints on the architectural design; especially reusability of software components in platform-independent algorithmic concepts are prevailing.
Conclusion: Software frameworks like the robotic operating system (ROS) enable fast prototypical solutions; however, architectural support for resource-constrained devices is limited. Here, architectural design drivers as realized in AUTOSAR are more suitable.

1 Introduction

Competitions like the 2007 DARPA Urban Challenge [RBL+08], the Grand Cooperative Driving Challenge in 2011 and its announced successor in 2016 [AEE+11, Hig11], as well as successful large-scale test runs on public roads [Thr10] led to the announcement from all important automotive OEMs that self-driving vehicular technology is said to be available for customers around 2020 [Hir13]. Since the technology has proven its feasibility on urban roads like in Braunschweig, Germany as reported by [NHO+11] or in the VisLab Intercontinental Challenge (VIAC) [BBFZ11], the next technological challenges are robustness, reliability, and foremost resource-efficiency.

1.1 Motivation and Problem Domain

Today's prototypical cars use either a broad variety of different sensors or depend on expensive and sophisticated sensors that can perceive a huge data volume from the surroundings so that algorithmic solutions have much information available to derive reasonable and reliable driving decisions. However, such large, clearly visible, and predominantly expensive sensors are rather unrealistic being part of a self-driving car for the mass-market.

Therefore, today's technological solutions need to become more resource-efficient in terms of realizing (at least) the same robustness and reliability as the existing self-driving vehicles with their broad and expensive selection of sensors by using less, cheaper, and simpler sensors [NP14]. Thus, limitations for the available hardware resources need to be carefully addressed also in the software architecture for self-driving cars.

1.2 Research Goal and Research Questions

Therefore, the research goal for this study is to identify and elaborate on architectural design drivers for self-driving vehicular technology that is subject to resource constraints. Thus, the following research questions were derived:

RQ-1: *Which design drivers affect the software architecture of self-driving cars when resource constraints need to be obeyed?*

RQ-2: *Which hardware- and software-related architectural design decisions can be found in successful self-driving cars?*

RQ-3: *Which experiences can be drawn from a competition of self-driving miniature cars regarding resource constraints and their impact on the software architecture for self-driving cars?*

1.3 Contributions of the Article

The contributions of this work include (a) an elaboration of the impact of design decisions originating from resource-constrained environments on the software architecture of self-driving vehicular technology by outlining results from a systematic literature review, (b) a selected overview of architectural design decisions for the hardware and software architecture of recent successful self-driving cars, and (c) transferring the conclusions drawn from the software architecture of self-driving miniature cars to real-scale cars regarding the software design for resource-constrained environments.

1.4 Structure of the Article

The rest of the article is structured as described in the following: In Sec. 2, an overview of related work is presented with a specific focus on resource-constrained self-driving vehicular technology. Sec. 3 describes the results from a multiple case study that we conducted using four self-driving cars projects including two projects for the development of self-driving miniature cars under resource-constraints. Sec. 4 discusses the outcome of the multiple case study, before the article concludes in Sec. 5.

2 Related Work

To address *RQ-1*, we conducted a systematic literature review (SLR) according to the guidelines provided by Kitchenham and Charters [KC07]. The results are described in the following.

2.1 Data Sources and Search Strategy

We used the following digital libraries as data sources for the SLR: ACM Digital Library (ACM DL), IEEE Xplore Digital Library (IEEE Xplore), and SpringerLink. Additionally, we used Google Scholar to complement the aforementioned data bases to also cover data sources, which are not represented in the aforementioned libraries. Within these databases, the following search strategy was applied:

Publication date: "2004 or newer" to ensure that related work might be influenced by the outcome for the 2004 DARPA Grand Challenge.

Search phrase A: "resource-constrained" AND "self-driving" and "car"

Search phrase B: "resource-constrained" AND "self-driving" AND "vehicular" AND "technology"

2.2 Study Selection

After the results were retrieved from the databases mentioned before, the following *exclusion criteria* were applied: (a) The publication was not inside the desired time frame (E_1), (b) the work was a duplicate (E_2), or (c) the title and abstract were referring to an unrelated domain (E_3). The remaining publications that are related to this article according to title and abstract are listed in the last column in Tab. 1.

Search phrase	Library	#	E_1	E_2	E_3	$\surd$	References
A	ACM DL	1	-	-	1	-	
A	IEEE Xplore	25	-	1	19	5	[JKK+14] [KKLR13] [KBRJ12] [CQB+12] [UW08]
A	SpringerLink	3	-	-	2	1	[FGRS14]
A	Google Scholar	20	1	$5^{1,2,3,4}$	11	3	[Kre09] [WLM13] [ME10]
B	ACM DL	0	-	-	-	-	
B	IEEE Xplore	0	-	-	-	-	
B	SpringerLink	0	-	-	-	-	
B	Google Scholar	4	-	1^{5}	1	2	[GLPL14] [Chu13]
	Total	53	1	7	34	11	

Table 1: Results from the systematic literature review.

2.3 Study Quality Assessment

Kitchenham and Charters state different risks for bias that might occur when conducting an SLR. To address publication bias, which refers to the risk that only papers are considered that report about positive results, we complement the well-known libraries by including Google Scholar as an additional data source.

The risk of performance bias refers to the dependency of the results on the researcher who is carrying out the SLR. Since the amount of results that we retrieved from the databases was surprisingly small, no specific means were applied to address this risk. Instead, all papers that are relevant to this study are summarized in the following.

2.4 Data Extraction and Synthesis

The remaining eleven papers are summarized and their relevance to this study is discussed.

Jo et al. [JKK$^+$14] describe in their work a distributed system architecture for self-driving cars that relies on AUTOSAR and FlexRay. The motivation for the former decision were "reusability, reliability, transferability, and maintainability of the software.", while for the latter, predictability in such a safety-critical system is required. However, the authors also conclude that the existing AUTOSAR architecture contains many constraints that limit the flexibility, which is required to successfully realize a self-driving car. Thus, they omitted "many standard AUTOSAR components related to product reliability."

Kim et al. [KKLR13] proposed a thread-based parallelization to realize the motion planning in self-driving cars to evaluate potential paths that can be considered as drivable. However, they do no outline design constraints for a resource-efficient realization of a self-driving car.

In [KBRJ12], Kim et al. present the middleware SAFER, which they applied to the winning self-driving car "Boss"; Urmson and Whittaker [UW08] give a brief overview of this vehicle. While the middleware aims for fault-tolerance, no design considerations are presented and discussed regarding resource-constraints.

Chong et al. [CQB$^+$12] present briefly their self-driving vehicle. Due to the very short article, no design considerations are presented or discussed.

In the doctoral thesis from Krenn [Kre09], the system architecture for an adaptive system for intelligent systems is discussed. The main focus is not on resource-constrained software architectures for self-driving cars.

Wang et al. [WLM13] describe an architecture for the ontology-based integration of components in the context of the " Internet of the Things" (IoT). In the main focus of the work is the handling of restrictions in terms of heavy communication layers for resource-constrained devices like RFID tags. However, specific considerations for the software architecture of self-driving cars are not presented or discussed.

[1]One article is a duplicate to the one found with the second search query in Google Scholar; article was not accessible.

[2]Two articles are duplicates to the ones found with the first search query in SpringerLink.

[3]One article is a duplicate to the one found with the first search query in ACM DL.

[4]Includes a duplicate to [Chu13].

[5]The article is the duplicate to the one found with the first search query; article was not accessible.

Fortino et al. [FGRS14] discuss requirements for smart environments as needed by the IoT. The authors have identified the following requirements, which they evaluated for different IoT-capable middlewares like the robot operating system (ROS):

1. "Abstraction over heterogeneous input and output hardware devices,"
2. "Abstraction over hardware and software interfaces,"
3. "Abstraction over data streams (continuous or discrete data or events) and data types,"
4. "Abstraction over physicality (location, context),"
5. "Abstraction over the development process."

Medvidovic and Edwards present a roadmap for "software architecture and mobility" [ME10]. In their work, they also cover some of the frameworks presented by Fortino et al. for mobile devices like smart phone but also for mobile robotic platforms. In their research roadmap, they confirm that the software engineering is specifically challenged by resource-constrained devices. While they suggest the use of generative techniques to create systems that are "correct by construction", some restrictions in terms deployment costs need to be kept in mind when dealing with such devices. Their observation is in line with the one from Jo et al. [JKK$^+$14] as described before. These design considerations can also be found in AUTOSAR. However, they do not discuss specific requirements that originate from resource-constrained devices.

Gerla et al. [GLPL14] report the "vehicular cloud" that combines independently acting autonomous vehicles with a cloud-based infrastructure. While they are envisioning various use cases in combination with vehicle-to-X communication, specific design aspects from resource-constrained devices are not discussed.

Chua focuses in his thesis [Chu13] mainly on unmanned aerial systems that are used in urban search and rescue environments. While specific aspects are discussed regarding resource-constraints, no reflections are presented how this should be considered in the design of a software architecture.

3 Multiple Case Study

For addressing *RQ-2*, a multiple case study was conducted covering four projects for self-driving cars. We are reporting this study according to Runeson and Höst [RH08].

3.1 Method

The objective for this multiple case study is to evaluate design considerations of the software architecture from different self-driving cars. In this regard, a specific focus is given on how resource-constraints are considered in the architectural design.

These architectural designs for self-driving cars are evaluated according to the criteria that have been identified from the SLR as provided by Fortino et al. [FGRS14]:

1. “Abstraction over heterogeneous input and output hardware devices,”
2. “Abstraction over hardware and software interfaces,”
3. “Abstraction over data streams (continuous or discrete data or events) and data types,”
4. “Abstraction over physicality (location, context),”
5. “Abstraction over the development process.”

The following cases are presented and compared in the multiple case study:

1. A ROS-powered platform for autonomous driving evaluated by a company as described by Ainhauser et al. [ABH$^+$13],
2. a ROS-powered platform for autonomous driving evaluated by a research laboratory as outlined by Fernandes et al. [FSP$^+$14],
3. results from a Java-based design of a self-driving miniature car that participated in the international competition for self-driving miniature vehicles CaroloCup[6],
4. results from a C++-based design of a self-driving miniature car that participated also in this competition [BAH13].

3.2 Case Description

The first architecture in our case study is described by Ainhauser et al. [ABH$^+$13], where ROS as a platform for future autonomous driving functions is discussed. The authors report that for autonomous driving two environments are needed: A hardware-oriented environment for sensors and actuators, as well as an environment for functions like connectivity and route planning. For this, the two frameworks AUTOSAR and ROS are used: AUTOSAR as the standard for the embedded hardware and ROS for high-level functions like communication, to build the environment model, and planning the route. AUTOSAR components are integrated by ROS stubs. They suppose that ROS is currently the most suitable existing middleware for the needs of autonomous driving. They propose the architecture in Fig. 1 on the left hand side.

The second case study is taken from a very recent publication in the Journal of Systems Architecture published by Fernandes et al. [FSP$^+$14], where two outdoor intelligent vehicles platforms named CaRINA I and CaRINA II were described. Similarly as in the first case, an architecture based on ROS is proposed. Since the authors have developed the architecture for the first vehicle and could transfer it to the second one, they suggest it as a reference architecture. Fig. 1 shows the architecture on the right hand side.

The third case study is taken from team “piCar” of the Technische Universitaet Braunschweig, that won the Junior Edition at CaroloCup 2014 with their car. The vehicle uses the Eleanor Framework, which was developed at the Institute of Software Engineering and

[6]www.carolocup.de

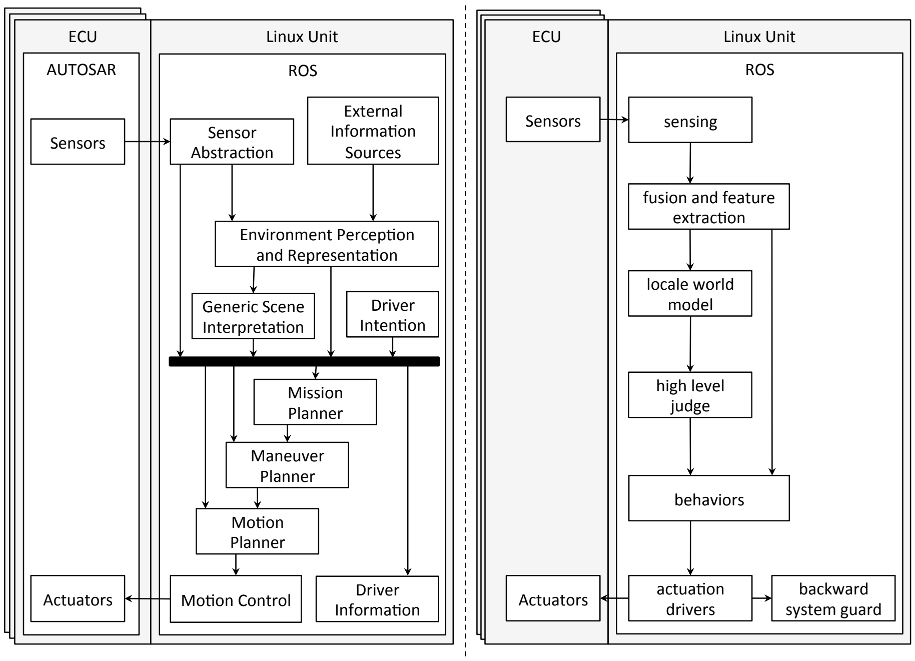

Figure 1: Comparison of the architecture that uses ROS as software platform for the self-driving car: On the left hand side, the industrial study is shown in comparison with the research laboratory realization as depicted on the right hand side.

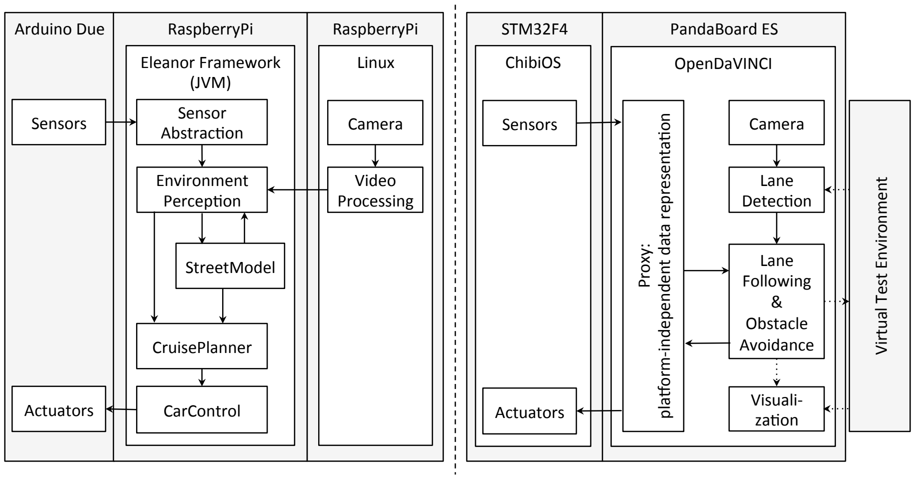

Figure 2: Comparing the architectural design of two resource-constrained self-driving miniature cars: On the left hand side, the architecture from team from Technische Universitaet Braunschweig is shown and on the right hand side, the design of the team from Chalmers | University of Gothenburg is shown. Both teams participated in the international competition CaroloCup.

Automotive Informatics. The architecture of the vehicle is shown in Figure 2 on the left hand side.

The vehicle uses an Arduino microcontroller to control sensors and actuators as well as two RaspberryPis. One of them acts as a central processing unit for route planning and the other is used to connect the camera and for video processing. The Eleanor Framework is a Java-based framework designed specifically to meet the needs of limited hardware, however, yet to offer hardware independence and extensibility.

The framework provides a service and extension registry inspired by OSGi[7], so all functions are implemented as plugins that can have dependencies among each other. It also provides useful basic functionality such as the abstraction of sensors and actuators as well as a thread manager to schedule the plugins. In contrast to the other cases, this system uses inter-process-communication (IPC) between the plugins.

The last case study is also taken from a team that participated twice in the CaroloCup and that won the Junior Edition at CaroloCup 2013. The architecture used in cars of Chalmers | University of Gothenburg is shown in Fig. 2 on the right hand side. The team used the simulation and development middleware OpenDaVINCI in combination with the real-time operating system ChibiOS/RT[8] as hardware abstraction layer (HAL) on the embedded system, which is realized by using an STM32F4 board. In this regard, this case shows similarities with the first case, where AUTOSAR is used as HAL. On the high-level data processing where the self-driving functionality is realized with C++, platform-independent data structures are used to enable reusability in the virtual test environment [BCH$^+$13].

4 Analysis and Discussion

To address *RQ-3* we compared the four case studies, to extract differences and commonalities. We have found the following results:

All four cases have a strict separation between low-level functions to control the hardware and a high-level layer for environment perception and behavioral planning. The separation is achieved by a distribution of functions to different hardware. Hardware-related functions were deployed to microcontrollers, where the first and fourth case use a hardware abstraction layer (AUTOSAR and ChibiOS/RT) and the third case implements the functions directly on the Arduino; no information was available for the second case.

On the high-level, all cases uses an abstraction layer over the hardware devices, which is in line with the criteria identified by Fortino et al. [FGRS14]: Case 1 uses ROS stubs for AUTOSAR components, case 2 uses modules that encapsulate the low-level protocols for each hardware component, case 3 uses a plugins that communicates with the hardware part and provides high-level services for other plugins, and case 4 uses a proxy that transparently maps platform-independent data structures to platform-specific representations that depend on the concrete hardware environment.

[7]www.osgi.org
[8]www.chibios.org

Due to the abstraction of the concrete hardware, (a) high-level algorithms such as route planning can be developed independently of the hardware, and (b) the hardware may be replaced without the need to change to high-level software. This is an important aspect since the configuration of commercial-of-the-shelf components, which are used as software/hardware-interface in resource-constrained environments like the CaroloCup competition, might change. Thus, even the low-level software that handles the data from sensors and to actuators can be reused, when an appropriate HAL like AUTOSAR or ChibiOS/RT is used.

The abstraction over hardware and software interfaces in the first two cases is realized with the infrastructure and facilities provided by ROS, where the modules are loosely coupled through topic publishing/subscription realized via TCP or UDP communication. The third case uses a service registry, where plugins can register their interfaces to exchange data. Case 4 uses a UDP-multicast session for a loosely coupled communication. As all cases uses a concept of a platform-independent data representation, there is an abstraction over data streams and data types as suggested by Fortino et al. [FGRS14].

As the first two cases use ROS that supports a distributed architecture, there is an abstraction over physicality; a similar concept is realized for case 4. Case 3 has a high level of abstraction over the context by using a Java virtual machine but there is no direct support for IPC.

As the development environment usually does not match the hardware on the vehicle, the compilation of the code has to be done on the limited hardware of the vehicle, which can be very time consuming, or by using a cross compiling tool chain. So, the development process of the low-level part depends on the hardware in-use. It can be simplified by using a HAL like AUTOSAR in case 1 or ChibiOS/RT in case 4. The development process of the high-level part in case 1 and 2 is given by the tool chain of ROS and the one of case 4 is set by OpenDaVINCI. As case 3 uses Java, any Java-based development process can be used. Here the usage of Java has the advantage that the code can be compiled on fast development machines and deployed on any hardware supported by Java.

We found that all four cases are very similar. Each uses a central processing unit for the high-level part with the responsibility for communication, creating an environment model, planning the driving route, and decide which maneuver to drive. Sensors and actuators are always connected through an ECU, which is abstracted by a low-level HAL in two cases.

Furthermore, the cases show that the low-level data processing can be realized with resource- and energy-efficient and hardware environments to handle the data from sensors and to actuators. A prevailing design consideration is the HAL which not only allows reusability of software components on that layer; moreover, it also enables code-generation of the software components due to standardized APIs.

For the higher levels, more processing power is reported to be required – from the results of the SLR but also to handle mass-data from cameras or to interpret and derive driving decisions. On this layer, reusing the self-driving functionality realized by the software components is of interest – for instance by embedding them in virtual test environments for example. A specific design aspect that originates from the resource-constrained low-level hardware environment is the mapping of platform-independent data structures to the

	Case 1	Case 2	Case 3	Case 4
separation between low/high-level functions	+	+	+	+
hardware abstraction layer	+	?	-	+
high-level abstraction layer	+	+	+	+
loosely coupled components	+	+	+	+
platform-independent data representation	(+)	(+)	+	+
platform-independent development process	-	-	+	-
energy-efficient low-level data processing	+	+	+	+

Table 2: Comparison of architectural design drivers.

concrete platform-specific representation. Our findings are summarized in Tab. 2.

In the following, we are reporting about threats to validity according to Runeson and Höst [RH08]. Considering construct validity, the setup of the study by using a systematic literature review first to investigate the field followed by a multiple case study is appropriate to investigate design criteria for resource-constrained self-driving cars. The use of the SLR also reduces the risk of reliability bias since the evaluation criteria originate from the findings of the SLR.

Regarding external validity, we analyzed four different cases by using criteria that are reported by literature. Due to the recent publication dates of the investigated first two cases as well as the involvement in the design and realization in the last two cases, the findings can be generalized to design criteria for self-driving cars with a focus on resource-constraints.

Additionally, the specific design of a multiple case study reduces the bias regarding internal validity because four different cases are investigated that were driven by four independently working research groups.

5 Conclusion and Outlook

Self-driving cars are said to be part of our future mobility. In the last decade, the research in this field has facilitated the development and state-of-the-art so that this technology is evaluated nowadays in large-scale on public roads. However, to bring the technology to the mass-market, the hardware costs especially for sophisticated sensors that are mounted on the vehicle's roof need to be scaled down.

In this study, results from a systematic literature review are presented that evaluate how resource-constraints are considered in the design of the software architecture of self-driving cars. This literature study unveiled that common guidelines could not be identified on how to handle such constraints accordingly. Therefore, a multiple case study was conducted to evaluate four projects for self-driving cars regarding design considerations for resource-constraints.

Common design criteria include abstractions from concrete hardware environments. While such a design does not only enable the independence from a concrete hardware, it also allows the transparent software reuse in virtual test environments for example. As future work, specific design criteria for platform-independent data models in combination with automated scenario generation for such test environments are of interest. However to operate autonomous vehicles on public roads, the legal framework in different countries needs to be evolved as well.

References

[ABH+13] Christoph Ainhauser, Lukas Bulwahn, Andreas Hildisch, Stefan Holder, Olexiy Lazarevych, Daniel Mohr, Tilmann Ochs, Michael Rudorfer, Oliver Scheickl, Tillmann Schumm, and Felix Sedlmeier. Autonomous Driving Needs ROS (ROS as Platform for Autonomous Driving Functions), May 2013.

[AEE+11] Bruno Augusto, Alireza Ebadighajari, Cristofer Englund, Usman Hakeem, Naga V Irukulapati, Josef Nilsson, Ali Raza, and Reza Sadeghitabar. Technical Aspects on Team Chalmers Solution to Cooperative Driving. Technical report, Chalmers University of Technology, Göteborg, 2011.

[BAH13] Christian Berger, Md. Abdullah Al Mamun, and Jörgen Hansson. COTS-Architecture with a Real-Time OS for a Self-Driving Miniature Vehicle. In Elad M. Schiller and Henrik Lönn, editors, *Proceedings of the 2nd Workshop on Architecting Safety in Collaborative Mobile Systems (ASCoMS)*, pages 1–12, Toulouse, France, September 2013.

[BBFZ11] Alberto Broggi, Michele Buzzoni, Mirko Felisa, and Paolo Zani. Stereo obstacle detection in challenging environments: the VIAC experience. In *IROS*, pages 1599–1604, 2011.

[BCH+13] Christian Berger, Michel Chaudron, Rogardt Heldal, Olaf Landsiedel, and Elad M. Schiller. Model-based, Composable Simulation for the Development of Autonomous Miniature Vehicles. In *Proceedings of the SCS/IEEE Symposium on Theory of Modeling and Simulation*, San Diego, CA, USA, April 2013.

[Chu13] Noon Heng Chua. *Integration of Multiple Unmanned Systems in an Urban Search and Rescue Environment*. Master thesis, Naval Postgraduate School, March 2013.

[CQB+12] Z J Chong, B Qin, T Bandyopadhyay, T Wongpiromsarn, B Rebsamen, P Dai, S Kim, M H Ang, D Hsu, D Rus, and E Frazzoli. Autonomy for Mobility on Demand. In *Proceedings of the IEEE International Conference on Intelligent Robotics and Systems*, pages 4235–4236, Vilamoura, Algarve, Portugal, October 2012.

[FGRS14] Giancarlo Fortino, Antonio Guerrieri, Wilma Russo, and Claudio Savaglio. *Internet of Things Based on Smart Objects*. Springer International Publishing, Cham, 2014.

[FSP+14] Leandro C. Fernandes, Jefferson R. Souza, Gustavo Pessin, Patrick Y. Shinzato, Daniel Sales, Caio Mendes, Marcos Prado, Rafael Klaser, André Chaves Magalhães, Alberto Hata, Daniel Pigatto, Kalinka Castelo Branco, Valdir Grassi, Fernando S. Osorio, and Denis F. Wolf. CaRINA Intelligent Robotic Car: Architectural design and applications. *Journal of Systems Architecture*, 60(4):372–392, April 2014.

[GLPL14] Mario Gerla, Eun-Kyu Lee, Giovanni Pau, and Uichin Lee. Internet of Vehicles: From Intelligent Grid to Autonomous Cars and Vehicular Clouds. In *Proceedings of the IEEE World Forum on Internet of the Things*, pages 241–246, March 2014.

[Hig11] High Tech Automotive Systems. Grand Cooperative Driving Challenge, January 2011.

[Hir13] Jerry Hirsch. Self-driving cars inch closer to mainstream availability, October 2013.

[JKK$^+$14] Kichun Jo, Junsoo Kim, Dongchul Kim, Chulhoon Jang, and Myoungho Sunwoo. Development of Autonomous Car â Part I: Distributed System Architecture and Development Process. *IEEE Transactions on Industrial Electronics*, 0046:1–10, April 2014.

[KBRJ12] Junsung Kim, Gaurav Bhatia, Ragunathan Rajkumar, and Markus Jochim. SAFER: System-level Architecture for Failure Evasion in Real-time Applications. In *Proceedings of the 33rd IEEE 33rd Real-Time Systems Symposium*, pages 227–236. Ieee, December 2012.

[KC07] Barbara Kitchenham and Stuart Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical report, Keele University and Durham University Joint Report, Durham, UK, July 2007.

[KKLR13] Junsung Kim, Hyoseung Kim, Karthik Lakshmanan, and Ragunathan Rajkumar. Parallel Scheduling for Cyber-Physical Systems: Analysis and Case Study on a Self-Driving Car. In *Proceedings of the 4th International Conference on Cyber-Physical Systems*, pages 31–40, Philadelphia, PA, USA, April 2013.

[Kre09] Willibald Karl Krenn. *Self Reasoning In Resource-Contrained Autonomous Systems*. Phd thesis, Graz University of Technology, 2009.

[ME10] Nenad Medvidovic and George Edwards. Software architecture and mobility: A roadmap. *Journal of Systems and Software*, 83(6):885–898, June 2010.

[NHO$^+$11] Tobias Nothdurft, Peter Hecker, Sebastian Ohl, Falko Saust, Markus Maurer, Andreas Reschka, and Jürgen Rüdiger Böhmer. Stadtpilot: First Fully Autonomous Test Drives in Urban Traffic. In *Proceedings of the International IEEE Conference on Intelligent Transportation Systems*, pages 919–924, Washington, DC, USA, October 2011.

[NP14] Paul Newman and Ingmar Posner. Robotics Science For Smarter Cars, May 2014.

[RBL$^+$08] Fred W. Rauskolb, Kai Berger, Christian Lipski, Marcus Magnor, Karsten Cornelsen, Jan Effertz, Thomas Form, Fabian Graefe, Sebastian Ohl, Walter Schumacher, Jörn-Marten Wille, Peter Hecker, Tobias Nothdurft, Michael Doering, Kai Homeier, Johannes Morgenroth, Lars Wolf, Christian Basarke, Christian Berger, Tim Gülke, Felix Klose, and Bernhard Rumpe. Caroline: An Autonomously Driving Vehicle for Urban Environments. *Journal of Field Robotics*, 25(9):674–724, September 2008.

[RH08] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, December 2008.

[Thr10] Sebastian Thrun. What we're driving at, April 2010.

[UW08] Chris Urmson and William Whittaker. Self-Driving Cars and the Urban Challenge. *IEEE Intelligent Transportation Systems*, 23(2):66–68, March 2008.

[WLM13] Wei Wang, Kevin Lee, and David Murray. Building a Generic Architecture for the Internet of Things. In *Proceedings of the 8th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 333–338, April 2013.